E X P E R I M E N T A T I O N:

Engine for Applied Research and Technology Transfer in Software Engineering

Dieter Rombach

University of Kaiserslautern
Computer Science Department
Software Engineering Chair
Kaiserslautern, Germany
&
Fraunhofer Institute for Experimental Software Engineering (IESE)
Kaiserslautern, Germany

Abstract: The empirical work in NASA's Software Engineering Laboratory in the 70's and 80's has contributed significantly to the maturing of the sub-discipline of 'experimental software engineering'. The development of experimental technologies ranging from the GQM approach for measurement to the EF approach for organizational learning provided the scientific basis; the successful experiments within the SEL development environment served as successful reference examples for others. The Fraunhofer Institute for Experimental Software Engineering (IESE) was founded in Germany based on the successful SEL principles. It was charged with speeding up the transfer of innovative software engineering technologies into a wide variety of industry sectors. The concepts of experimentation were developed further and used for a wide range of purposes from applied research to technology transfer and training. Already during the short history of IESE a successful track record of transferring innovative technologies fast and with sustained success has been established. This presentation focuses on the adaptation of the successful SEL concepts to a different environment, surveys the wide range of applications of 'experiments' as engine for successful technology transfer in a human-based development environment, and predicts a growing importance of experimental work in the future.

1. **Motivation.** The software domain can be characterized by two major facts: (1) The gap between state-of-the-art as taught at universities and state-of-the-practice as 'lived' in most commercial software development environments is_significantly higher than in other

engineering domains, and (2) the body of knowledge available to practitioners consists predominantly of technologies (e.g., languages, techniques, and tools), rather than methods and knowledge regarding the effects of such technologies in practical development environments. One conclusion is that progress in practice is not hindered by lack of technology, but by lack of such latter knowledge which hinders the transfer into practice. Let's just illustrate the problem for one example technology: There exists a very large number of testing techniques today. However, little knowledge exists as to the relative strengths and weaknesses of these techniques in different industrial settings. So, why would a project manager decide to use an alternative testing technique as opposed to the one in use for several years. What is needed can be compared best to so-called 'engineering handbooks' in other engineering disciplines. Such handbooks describe the available technologies together with their applicability, strengths and weaknesses for different constraints. This paper describes how such knowledge can be accumulated in a human-based development environment via experimentation.

2. **Experimentation.** There exist many different ways of accumulating software development knowledge. One very important form of such knowledge is experience derived from actual application of technologies. That means experience is based on product/process feedback loops in that process technology is applied, the impact on the resulting products is observed, and possible improvements regarding the process technology are identified via root cause analysis. In the context of this paper, experience resulting from projects accidentally or experience existing implicitly only is not considered. However, all experiences resulting from systematic hypothesis testing in either fully controlled laboratory experiments or semi-controlled field experiments and field case studies, and producing explicitly sharable insights (models) are considered. Experiments are one of the pre-requisites for sustained learning; it is much easier to change behavior based on documented first-hand experience, rather than knowledge from the world-at-large. Experiments are applicable to basic research for the purpose of understanding, to applied research for the purpose of packaging technologies together with information about their effects in varying project contexts, to teaching & training in order to experience the benefits of new technologies for one's own development tasks before project pressure could result in falling back

to the old technologies for the fear of risk regarding one's own performance, and to technology transfer for the purpose of adapting new technologies optimally to one's project context and providing cost/benefit.

3. **The Role of Experimentation in Software Engineering.** The software domain is characterized by a number of specific characteristics. The most important ones are that most development technologies are human-based and that the data are less frequent and mostly of non-parametric nature. The human-based nature of most technologies makes (a) the change process particularly hard as the 'execution engine' human being needs to be convinced of the benefits of changing to a new technology, and (b) the success of any new technology depends on the adherence to the process guidelines associated with that new technology. Both involves weighing the risk of using the new technology versus the risk of staying with the old technology. Basically, the cardinal question is 'Does it work for ME?'. Experience data from one's very environment are an important source of confidence for changing to and staying with a new technology. The less frequent and mostly non-parametric nature of software engineering data requires different techniques for data analysis – especially the combination of qualitative and quantitative analysis. Beyond that, many of the experimental techniques known from other areas can be applied.

4. **Available Tool Box for Experimental Software Engineering.** The existent body of technologies for experimentation in software engineering itself is significant and growing constantly. Most of the techniques have been initially created in (or have been at least stimulated by) NASA's Software Engineering Laboratory (SEL). Among the most important technologies are

   - the Goal/Question/Metric (GQM) approach for measurement (e.g., [Bas93.1], [Rom91]), supporting the derivation of metrics from a comprehensive goal specification
   - the Quality Improvement Paradigm (QIP) method (e.g., [Bas93.2]), enabling the integration of sound project feedback for project control with cross-project learning (NOTE: It adapts the Plan/Do/Check/Act approach from manufacturing to the specifics of the software domain)

- the Experience Factory (EF) approach (e.g., [Bas93.2]), defining extra learning related roles and integrating them with the traditional software development roles
- a portfolio of experimental designs (e.g., [Bas86]), ranging from controlled experiments to regular field case studies
- a variety of analysis methods (e.g., [Bri92]) for non-parametric software engineering data, integrating qualitative and quantitative analysis techniques

In addition, there exist

- a number of reference laboratory environments applying the above experimental technologies such as NASA's SEL as the 'mother of all laboratory environments', Fraunhofer IESE in Germany, and CAESAR in Australia
- a number of exchange forums such as the International Network for Software Engineering Research (ISERN) for researchers or the Software Experience Consortium (SEC) for practitioners
- a growing number of conferences (e.g., METRICS , SEL Workshop) and journals (e.g., International Journal for Empirical Software Engineering)

All this provides a sound starting point for experimental work. The ISERN Network is open to everybody interested in further developing the experimental technologies, teaming up in concrete technology experiment replication, and exchanging all kinds of experiences. The contact address is 'isern@informatik.uni-kl.de'. The SEC Consortium is open for application by companies active in the area of empirical work or corporate experience management. The contact address is 'fshull@fc-md.umd.edu'.

5. **Fraunhofer IESE: An Institute built on the Experimental Paradigm.** The Fraunhofer Gesllschaft e.V. in Germany is Europe's largest applied research and technology transfer organization. It consists of 48 institutes ranging in application domain from material sciences and production technology to information & communication technology and life sciences. These institutes receive approximately 30% base funding from government; the remaining 70% of their operating budgets have to be covered from industry project income.

The Fraunhofer Institute for Experimental Software Engineering (IESE) became the 48th permanent Fraunhofer Institute [Rom96]. Founded in 1996, its area of competence is software engineering; its applied research and transfer model is based on the experimental paradigm. That means Fraunhofer IESE helps companies to establish experimentally based learning organizations as a pre-requisite for sustained improvements, and then helps them introduce new innovative software development technologies (technical & managerial). With the base funding from government, technologies from basic research institutions are being evaluated via experimentation, and packaged together with the experimental results for transfer into specific domain and company environments.

Today, Fraunhofer IESE employees 80 full time scientists together with about 60 part-time personnel such as students and consultants. Th institute language is 'English'; 25% of personnel is non-German. The percentage of industrial income has risen to about 70% within three years. Collaborations include a large number of Europe's leading companies in the sectors of telecom, automotive & aerospace, and banking/insurance/trade.

Fraunhofer IESE has been created as the German instantiation of the NASA/SEL laboratory model. It was widely accepted that a closer collaboration between academia and industry was needed. This institutionalized model – allowing for long-term trusting relationships between academia and companies – was the answer. The reference to the working SEL example was one of the major arguments to finally convince companies and government of the opportunity at hand. Many of the concepts of IESE are based on SEL experiences by myself during my tenure at the University of Maryland and my involvement with NASA/SEL during the 1986-1991 time frame.

The main SEL concepts adopted include

- provision of an environment in which researchers, software developers, and customers can work together
- use of experimentation as a major research and technology transfer engine
- establishment of long-term relationships with development organizations

- exposing researchers to practice and practitioners to research
- have research being driven by practical needs (= applied research!)

However, there are some important differences compared to the SEL. They include

- operation as a business due to the fact that Fraunhofer Gesellschaft e.V. is a legal non-for-profit entity not associated with any university or for-profit company environment (business plan for 140 employees!)
- tougher sales job for close academia/industry collaboration due to a historically wider gap between academia and industry in Germany as compared to the US
- need for critical mass in IESE core competence areas personnel-wise due to the expectation by companies to support them strategically (i.e., long-term, always with experienced personnel)
- need for application sector know-how in addition to software engineering competence due to the fact that IESE collaborates with companies from different industry sectors
- need for complex incentive structure in order to provide equal motivation to researchers and practitioners working in IESE

Although, many of the experiences and lessons learned within the SEL could be reused, the changes due to the collaboration culture and heterogeneity in customer base posed the biggest challenges. However, the achieved high standing of IESE within the scientific and industrial community demonstrates the possibility of replicating the SEL experience.

6. Useful Applications. This section describes briefly some of the typical applications of the experimental paradigm within the Fraunhofer IESE. These applications comprise – due IESE's mission – applied research, teaching & training, and technology transfer. It is intended to describe the wide applicability and usefulness of experimentation – even in a very industry oriented setting.

6.1. Applied Research. It has been firmly established at IESE that applied research in software engineering produces new/refined/existing technologies together with recorded observations regarding their effectiveness in one or a class of industrial setting (i.e., certain

constraints). These observations need to be produced by some appropriate form of experimentation. These observations are only useful, if the underlying experiment is documented well enough to be repeatable by anyone challenging the findings or trying to replicate them in a slightly different environment. Observations from non-repeatable experiments do not contribute to the state-of-the-art. In that context, it must also be agreed that experiments with negative results are equally valuable. Negative results combined with qualitative analysis investigating possible causes and deriving new hypotheses contribute to learning. There exist only badly designed and/or performed experiments, no bad results!

Such experiments have been done for most of the IESE technologies ranging from software development to management and experimental technologies. The most prominent experiments include the

- effectiveness & efficiency of step-wise abstraction code reading (e.g., [Bas87])
- effectiveness & efficiency of perspective-based requirements reading (e.g., [Bas96])
- maintainability of well-structured OO programs (e.g., [Bri97])
- maintainability of well-documented (traceability from requiremens to code) programs
- cost/benefit ratio for product line development

All these experimental results are published in the literature. Most of them are accessible through the IESE web site. More experiments on the above as well as other topics are needed. Every software engineering researcher should feel challenged to participate. The International Software Engineering Research Network (ISERN) provides a stimulating environment to learn, share and collaborate. Please contact ISERN (www.iese.fhg.de/ISERN/, isern@informatik.uni-kl.de)!

6.2. **Teaching & Training.** Software engineering teaching and training must include the topic of experimental methods (see e.g., CMSC735 at the University of Maryland OR SE2 at the University of Kaiserslautern) as well as their practical application to self-experience important software engineering principles (see examples from the University of Kaiserslautern below!). The simple lecturing of software

engineering principles results too often in them being ignored during the next development tasks. Again the issue is that changing behavior requires motivation that the risk of change is manageable. Experiments as part of teaching can provide the necessary motivation. During practical industrial training such experiments can be repeated for the same reason of motivation for change. In addition, experimentation can demonstrate the applicability of some technology to the specific company setting and suggest some necessary adjustments prior to real use.

Together with the University of Kaiserslauternn Fraunhofer IESE has developed a number of technology demonstration experiments which are being repeated during every graduate level software engineering class as well as during industrial training (modified according to company constraints!). The standard experiments include

- demonstrating the superiority (i.e., effectiveness, efficiency) of code reading over unit testing (adaptation of the old 'Selby' experiment) (e.g., [Lot96])
- demonstrating the superiority (better understandability, modifiability) of well-structured OO designs over worse structured ones
- demonstrating the superiority (better modifiability) of tractably documented programs over less tractably documented ones
- demonstrating the superiority (i.e., effectiveness, efficiency) of perspective-based reading of informal requirements over other reading techniques

Each of these experiments has been performed at least three times. Comprehensive laboratory packages are available describing the experiment and providing key artifacts for easy replication in other environments.

6.3. **Technology Transfer.** The purpose of experimentation in technology transfer is twofold: First before the introduction of a candidate new technology experimentation helps to convince personnel (top management to invest in it, project management to support it, and project personnel to 'live' it under project pressure) of the potential benefits of a pre-packaged new technology, and it helps to adapt pre-packaged technology to specific needs of the target organization. Second during use of the new technology

experimentation helps to change the technology further in order to optimize its effects, and it helps to re-enforce its continued use and, as a result thereof, ensures its continued gains.

During its 3 year history Fraunhofer IESE has contributed to many sustained process improvements in industry which would have been impossible without experimentation (e.g., [Lai97]). An extensive list of company references can be obtained from the IESE web site.

7. **Outlook.** Experimentation is becoming an integral sub-discipline of software engineering. Reflecting the general needs of an engineering discipline and the specific characteristics of the software domain, a body of technologies and reference applications have been created. The role of NASA/SEL has been equally instrumental to the area of experimentation as has been the SEI's role to the area of assessments. NASA/SEL together with its off-springs (e.g., Fraunhofer IESE) has pioneered the application of experimentation to speed up the accumulation of shareable, testable & repeatable knowledge in research, to raise a generation of true software engineers thru teaching and training, and to speed up the infusion of innovative software development technologies into practice in technology transfer programs. More and more environments will recognize that experimentation does not represent additional effort, but rather speeds up the production of real contributions to the state-of-the-art in software engineering and their transfer into practice. As the performance of real experiments require laboratory set-ups at universities or in companies, more of such environments must be established.

   I wish the SEL a successful future! May it spin off more laboratory environments around the globe! May it be valued inside NASA as highly as it is outside!

8. **References.**

→ [Bas93.1] Basili, Caldiera & Rombach, The GQM Paradigm, in 'Encyclopedia of Software Engineering' (John J. Marciniak, Ed-in-Chief), John Wiley & Sons, Inc., 1993.

→ [Bas93.2] Basili, Caldiera & Rombach, The Experience Factory, in 'Encyclopedia of Software Engineering' (John J. Marciniak, Ed-in-Chief), John Wiley & Sons, Inc., 1993.

→ [Bas87] Basili & Selby, Comparing the Effectiveness of Software Testing Strategies, IEEE Transactions on Software Engineering, vol. 13, no. 12, pp. 1278-1296, December 1987.
→ [Bas86] Basili, Selby & Hutchens, Experimentation in Software Engineering, IEEE Transactions on Software Engineering, vol. 12, no. 7, pp. 733-743, July 1986.
→ [Bas96] Basili, Laitenberger, Shull et al, The Empirical Investigation of Perspective-Basded Reading, International Journal of Empirical Software Engineering, vol. 1, no. 2, pp. 133-164, 1996.
→ [Bri92] Briand, Basili & Thomas, A Pattern Recognition Approach to Software Engineering Data Analysis, IEEE Transactions on Software Engineering, vol. 18, no. 11, pp. 931-942, November 1992.
→ [Bri97] Briand, Bunse et al, An Experimental Comparison of the Maintainability of Object Oriented and Structured Design Documents, International Journal of Empirical Software Engineering, vol. 2, no. 3, pp. 291-312, 1997.
→ [Lai97] Laitenberger & DeBaud, Perspective-Based Reading of Code Documents at Robert-Bosch GmbH, Information & Software Technology, vol. 39, pp. 781-791, 1997.
→ [Lot96] Lott, Rombach, Repeatable Software Engineering Experiments for Comparing Defect-Detection Techniques, International Journal of Empirical Software Engineering, vol. 1, no. 3, pp. 241-277, 1996.
→ [Rom91] Rombach, Practical Benefits of Goal-Oriented Measurement, in Software Reliability and Metrics (Fenton & Littlewood, Eds.), Elsevier Publ., 1991.
→ [Rom96] Rombach et al, New Institute for Applied Software Engineering Research, International Software Process Journal, vol. 2, no. 2, 1996.

9. **Contacts**. For further information about this paper, please contact the author under ‘rombach@iese.fhg.de’. For further information regarding the Software Engineering Chair at the University of Kaiserslautern, please check ‘wwwagse.informatik.uni-kl.de’; for further information about the Fraunhofer Institute IESE, please check ‘www.iese.fhg.de’. For information about the International Software Engineering Research Network (ISERN), please check ‘www.iese.fhg.de/ISERN/’ or contact ‘info@iese.fhg.de’. For information about SEC Consortium, please contact ‘fshull@fc-md.umd.edu’.